DEVELOPMENT OF AN AUTOMATIC MONITORING PROGRAM FOR THE CONTROL DATA CORPORATION 1604 DATA PROCESSOR

BRICE L. BRADSHAW

DEVELOPMENT OF AN AUTOMATIC MONITORING PROGRAM
FOR
THE CONTROL DATA CORPORATION 1604 DATA PROCESSOR

* * * * * * * * *

Brice L. Bradshaw

DEVELOPMENT OF AN AUTOMATIC MONITORING PROGRAM FOR THE CONTROL DATA CORPORATION 1604 DATA PROCESSOR

by

Brice L. Bradshaw

Lieutenant, United States Navy

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE
IN
ENGINEERING ELECTRONICS

United States Naval Postgraduate School
Monterey, California

1 9 6 1

DEVELOPMENT OF AN AUTOMATIC MONITORING PROGRAM FOR THE CONTROL DATA CORPORATION 1604 DATA PROCESSOR

by

Brice L. Bradshaw

This work is accepted as fulfilling

the thesis requirements for the degree of

MASTER OF SCIENCE

IN

ENGINEERING ELECTRONICS

from the

United States Naval Postgraduate School

ABSTRACT

This paper is concerned with the development of a programmed, and hence computer controlled, automatic monitoring system for adaptation of a high-speed digital data processor, specifically the Control Data Corporation 1604, to simultaneous use as a radar data processor and in ordinary job-shop applications. This development is hopefully intended to serve two purposes: First, to demonstrate the feasibility of such an operation; and, Second, perhaps to enable use of the CDC 1604 at the United States Naval Postgraduate School as a radar data processor in a system currently under development by the Engineering Electronics Department as an instructional tool in modern radar digital techniques.

It is hoped that the effort expended may be of help to other 1604 users in the development of other similar programs.

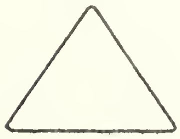
The writer wishes to express his sincere gratitude to Associate Professor Mitchell L. Cotton, United States Naval Postgraduate School, for his original suggestion and subsequent assistance, encouragement and cooperation in the preparation of this paper.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

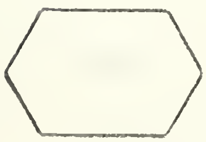
GLOSSARY OF SYMBOLS AND NOTATION

Functions performed by stored program steps

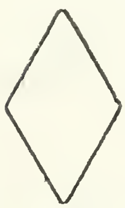


Decisions

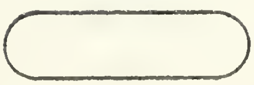
Flags or initial index settings



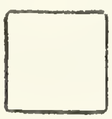
Return jump sub-routine or an auxiliary routine that acts in similar fashion



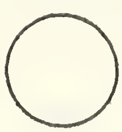
Indexing or counting function



Jump point



Wait



Identified program break and entrance points (also used to indicate continuity in flow charts when broken for page fit)

A-M	The automatic monitoring program, or Auto-Monitor, described in this paper
BCD	Binary coded decimal
m	Address field of one half of a word
M	An Address
(M)	Contents of address M
$M^u m$	Address field of upper half word of address M
$(M^u m)$	Contents of cell whose address is in $M^u m$
$(M^u m)^l$	Lower half of word whose address is in the m field of M^u
rA, rQ	The A register, and Q register
(rA), (rQ)	Contents of specified register

1. Introduction

This paper is the result of work begun in a course[1] in modern radar techniques emphasizing digital data processing and related topics. Since the Engineering Electronics Department has now in the advanced planning stage and early engineering development phase a prototype radar which will be used primarily as an instructional tool in the modern radar techniques of digital data processing, most discussion was slanted in the direction of application to this project. This planning includes a Control Data Corporation 160 Data Processor to be used as a digital detector and a display unit contracted to Data Display, Inc., of St. Paul, Minnesota. Because of the word length, memory capacity and attendant slow speed in handling problems of the magnitude envisioned, track correlation, up-dating and generation of display could not be carried on in the units whose availability was anticipated. For these functions, hereafter called radar processing, the services of a large scale, high-speed data processor would be needed. The Postgraduate School already possessed such a machine in the CDC 1604 Computer and it was known that the proposed 160 would be installed in a so-called satellite mode for rapid communication with the 1604, in a manner making it possible for both computers to operate on different facets of the problem with communication and control necessary for

[1] Es 449, U. S. Naval Postgraduate School, 2nd term, 1960-1961

exchange of information. Only one problem, philosophically speaking, remained. This was that using the 1604 for this purpose robbed the School of its use as a general purpose computer during those times it would be used in its radar processing capacity. It seemed advisable to attempt to do both at the same time.

Briefly, the problem is this:

a. Provide an interrupt or a scanning device in the 1604 at intervals to be determined.

b. Upon interrupt, perform the radar processing, then continue the scan or return to the previous program.

c. Bring programs into the computer, run them to completion and dump the desired data in the desired form for the user or customer.

The remainder of this paper is devoted to the consideration of these three statements and the resultant programming attack.

Section two is a description of the Auto-Monitor, or A-M for short, in its present form, with a discussion of the principal limitations. A table of the performance figures is included to provide a comparison with the final results and the original specifications.

Section three is a summary of the work done in the course mentioned previously, and presented in term paper[1] form. This work formed the point of departure to the work of this paper.

[1] Brice L. Bradshaw, "Auto-Monitor Routine." (unpublished term paper, U. S. Naval Postgraduate School, Monterey, Calif., 1960)

Section four is the history of the analysis, trial and error and research done in the development of the A-M.

Section five contains the conclusions as to the feasibility of the use of this program and conditions under which best utilization is realized. Some discussion is included to indicate areas in which immediate work may be done to increase its usefulness.

The flow charts of this program are included in APPENDIX A and APPENDIX B is essentially a short handbook of the use of the program.

Some flow charts of the interim phases of the A-M and notes on evolution are included in APPENDIX C for ready reference by the reader.

For those interested, a machine language program in AR format is included as APPENDIX D.

2. Auto-Monitor Program

The A-M in present form meets the considerations stated in the introduction. Figure 1 is a general block diagram of the basic relationships involved. Users' programs are read from magnetic tape in machine language. The programs are processed and output dumped at users' option in a machine language dump, a BCD Listable dump, or either Decofl or Glout may be programmed directly and used instead. After running and dumping the output of one program, the cycle is repeated. Each 1/60 second an interrupt occurs at which time an auxiliary routine is entered. This auxiliary routine makes it possible to perform the radar processing program and then return to the prior program. At present this radar function is simulated by a waiting routine of 6830 usecs, but only minor address changes will be necessary to incorporate genuine radar data processing. The exchange of data for the radar processing will be with a CDC 160 on the 1604 communication channels five and six. The return to the users' program is made following the radar processing if a pre-determined estimated running time has not been exceeded. If this running time has been exceeded, the A-M dumps the program and proceeds to the next program.

Input programs are pre-recorded on magnetic tape in machine language as mentioned above. A parameter record of five words precedes each program. Composition of the parameter record is explained

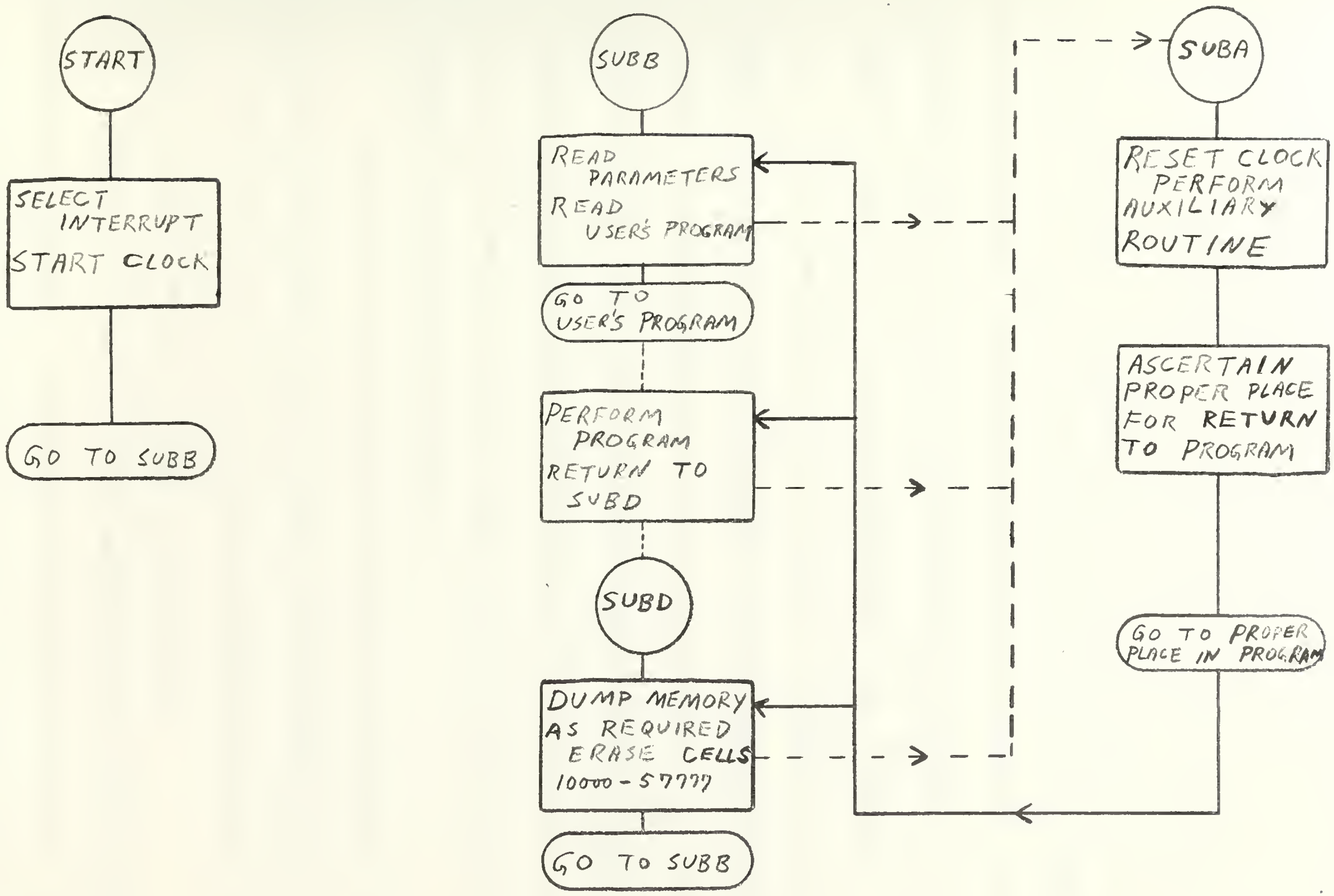


Fig. 1 - Block Diagram of Auto-Monitor Program

fully in APPENDIX B. The program itself follows the parameter record in one word records. Each program is followed by an end-of-file mark as an end marker and the last program on the tape is followed by two end-of-file marks to signify this fact. If a read length error occurs on parameter read-in or a zero address read as the starting address of the input program, a parameter read-in error is assumed and "PARAMETER ERROR" is printed as the dump for that program. No identification is printed and the program will not be run. Similarly, if a parity error occurs on parameter record read-in, or program read-in, the program will not be run and "PARER" is printed as the dump for that program. The identification may be any eight characters, including spaces, in BCD, and is one of the parameters required. After a successful program read-in, the identification is printed on the dump tape to identify the dump. All error signals and the identification are printed in BCD for listing on a line printer.

To allow for time consumed by radar processing, the users' estimate is multiplied by ten if longer than one second and by twenty if not. This corrected factor is used as the maximum running time to safeguard against endless looping in the program.

After a program has run to completion, the proper dump is enabled and executed. The present BCD Listable dump will require some modification for continuous use since it contains error stops, but

successful runs have been made using this dump, owing to the high reliability of the tape units. If a user has programmed his own dump, or used Decofl or Glout, no dump selection need be made. After the dump, "END DUMP" is printed on the dump tape. Cells 10000 through 57777 are then erased and the new program parameters read in to start the cycle anew. If the last program on the tape has been processed, an end-of-file mark will be read for the parameters and the tape will rewind to interlock to signify this fact to the operator.

Referring to Figure 1 again, the A-M consists of four parts: START, SUBA, SUBB and SUBD. The flow charts for these parts are collected and placed in APPENDIX A.

START, see Figure A-1, selects an interrupt on arithmetic overflow, and starts the real time clock. The clock functions to create a sign change in the Accumulator each 1/60 second. This sign change is sensed as an arithmetic overflow producing the interrupt. After starting the clock, exit is made to the TAPELKOUT subroutine of SUBA.

SUBA, see Figure A-2, is the auxiliary routine entered at the time of interrupt. Tape movements are monitored to determine subsequent action. Then a directed exit is made to the radar processing program and upon return to the action previously found necessary. If the return to users' program is to be made, the elapsed running time is checked. If it is excessive, then a jump to the dump routine is made.

SUBB, Figure A-3, is the sub-routine which reads programs into the computer with proper checks to insure accuracy of the programs. It completes the arrangements necessary for the running of the programs.

SUBD, see Figure A-4, selects the proper dump as directed in the parameters, executes it and then erases the cells 10000 through 57777. For this reason, it is advisable that programs to be run using the A-M be written for this range.

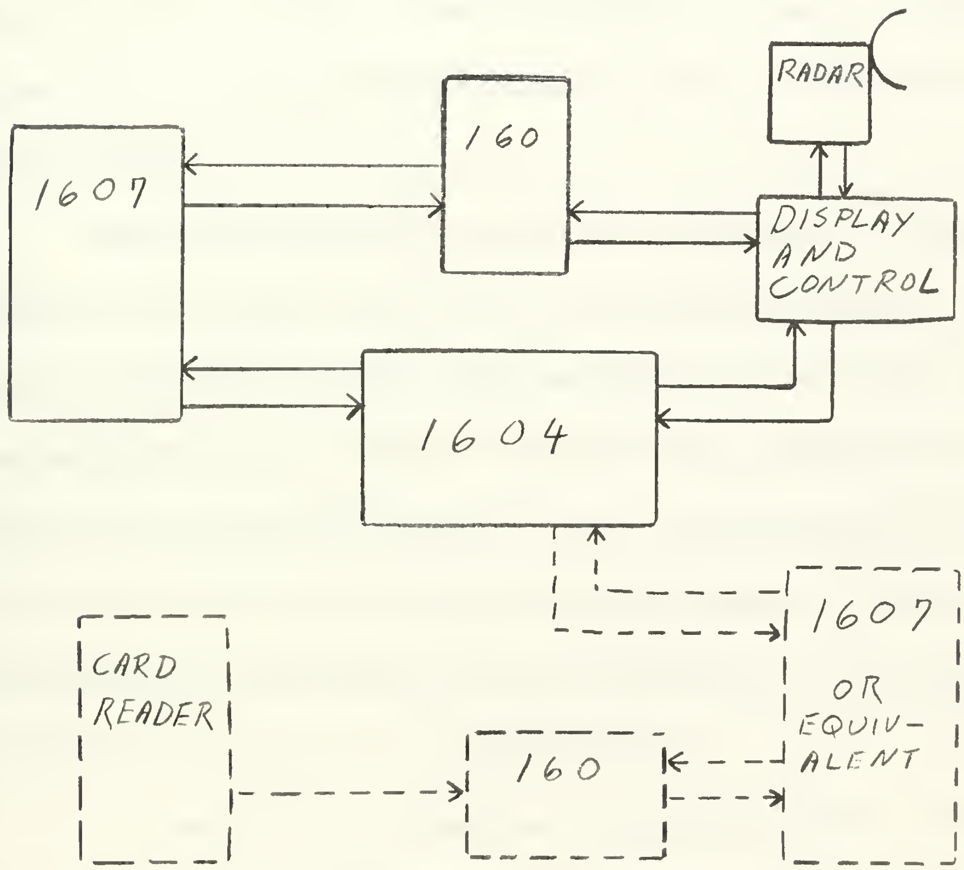


Fig. 2 - Equipment Relationship Controlled by Present A-M

Figure 2, above, is an indication of the anticipated use of this program. For the present ignore the dotted blocks. They are considered in Section five. The input tapes are read from a 1607 Magnetic Tape System connected to communications channels three and four and the dumps are made to the same unit. At present, input tapes are to be designated "Three" and output tapes designated "Four." If a LIB call feature is incorporated, it is anticipated that the library tape will remain designated "One." This makes it possible for a user to program inputs from "Two" in a normal fashion should he need to bring taped data into the computer.

After the completed tape "Three" has rewound to interlock, a "Tapelockout" feature may be activated and tapes changed or a new unit designated "Three." Deactivating the Tapelockout will then commence processing of the new "Three." Tapelockout is accomplished by putting jump key one down (OFF position). This locks the A-M away from the tape system and the user's program in the computer. Release is accomplished by raising jump key one (ON position) and results in a jump to SUBB for the start of a new read-in cycle.

If a tape is to be changed and the program in the computer is to be preserved, put the desired tape on "Stop manual" and the program will be stopped at the next point of use of that tape. When the tape is changed and the unit put in the rewound position, the program will continue at its proper place.

Referring to Figure 2 again, the 160 satellite system uses channels five and six for communication with the 1604. This makes it possible to communicate with the 160 while communicating with the 1607 containing the input and output program tapes with parity checks and other tape service features available. The above mode of communication was assumed for the development. Actual communication between the computers is via the DDI unit, the DD 65.

A routine has been written which will load programs in the proper format for the input in its present form.

The A-M has a provision for bootstrapping itself into the computer. This requires that it be placed on tape in machine language, of course.

Timing results of the A-M showed 0.8 msecs of A-M time per period. Each program word read in requires 8 msecs, and each dump tape movement requires 10 msecs. In addition, 200 msecs are required for erasure of computer memory cells after each program has been run. After the actual radar processing time is known, time for program runs may be computed readily. The following table summarizes the percentage of A-M dead time as a function of the interrupt period.

Period	Percentage dead time
16.67 msecs	4.8
33.34 msecs	2.4
50.00 msecs	1.6

100.00 msecs	0.8
250.00 msecs	0.32
333.33 msecs	0.24
500.00 msecs	0.16
1000.00 msecs (1 sec)	0.08

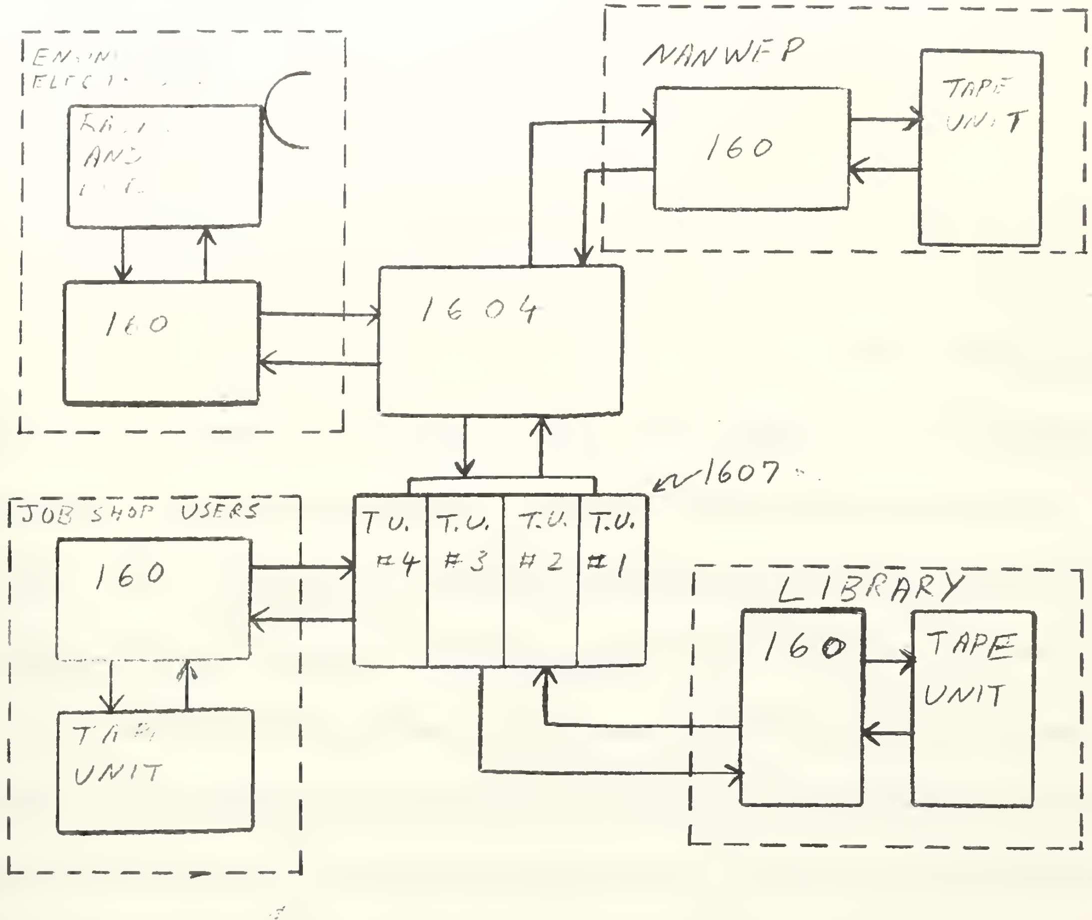


Fig. 3 - Broad Concept of Equipment to be Coordinated by the Auto-Monitor Routine

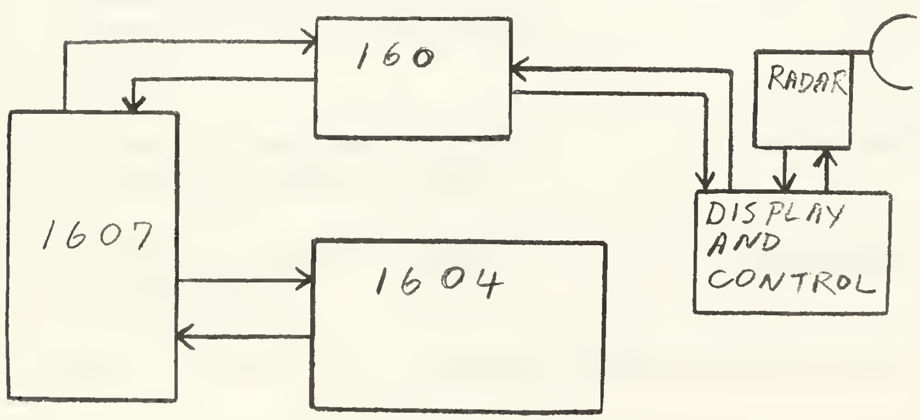


Fig. 4 - Block Diagram of Simplified Problem

3. Background

Figure 3 is the concept of the broad problem from which the study was started. This included four CDC 160's with two or more 1607 units in a satellite mode of operation with the 1604. It was thought that the 160's would be attached to the 1604 channel 7 for high speed transfer. These 160 computers would be used as the input for four users, i.e., Electronics Department (radar processing), the Numerical Weather Analysis group, the Postgraduate School Library and the job shop users of the 1604. The concept was to use the 1604 in a scanner mode, interrogating the equipments in order and performing the functions requested by each on a priority basis. The parameters picked as a starting point were:

Period of radar processing	Each milli-second[1]
Period of equipment scan (Look at each equipment and perform functions on priority basis)	Each second
Period for functions requiring excessive tape movement	Ten seconds
Percent utilization by radar processing	Approximately 50%

Obviously this is an extremely difficult problem, so, as is usual, the problem was immediately simplified as far as possible to a scope

[1] Selected from standpoint of possible use of 1604 as detector in radar processing system

susceptible to attack by one inexperienced programmer. Once feasibility had been demonstrated, and the basic problems and solutions uncovered, effort could be expended in sophistication to the degree thought practicable.

Figure 4 is the equipment relationship which was used in the simplified approach. The satellite system as actually installed utilized the 1607 system for communication instead of the channel seven as first supposed. Radar inputs from the 160 and job shop users' inputs from tape were assumed, both of these through a single 1607.

The 1604 has a built-in real time clock of period 1/60 second with a programming feature to provide program interrupts at multiples of this period. The basic 1/60 sec period was selected because this was the shortest one above the one msec original parameter. This makes it possible to process 240 tracks for a 15 r.p.m. antenna scan rate, processing one track on each interrupt.

Each second a scan of the parameters for a tape input was made to determine if there was another program on tape to be brought into the computer. This fact was flagged for the computer to note when the present program had been finished. If parameters for program input were available, the program would be read in to the computer, run to completion and then dumped as required.

Because of the obvious damage that an error made in either program or parameter read-in would do to a program attempting to operate continuously, checks had to be made to assure accurate tape read-in.

In ordinary programming this presents no problem. Here, however, should an interrupt occur while a buffer was in progress, and the communication channel three used by the radar processing program before the parity error check was made, the parity error information would be lost.[1]

Since timing relationships were unclear at this time, it seemed advisable to provide for the radar processing routine's extending to the next interrupt cycle. This necessitated returning to cell 00007 to remove the interrupt lockout so that the next interrupt would be effective. If an interrupt occurs in the upper half of an instruction, return is made to the lower half of the instruction and the information making this possible is stored in a flip-flop utilized in the return to cell 00007.

To solve these two problems, a programming restriction was assumed, i.e., the order, "buffer activate," "wait to finish of buffer," and "sense parity error" had to be followed in the program read-in. Then, after each interrupt, unless the interrupt occurred in the radar processing routine, the program was altered ahead of the interrupt address so that a jump to the interrupt routine was made. The program was then entered via cell 00007 and when the altered instruction returned to the interrupt routine, the original instruction had to be replaced. In order to obtain parity error checks, the instruction replaced was the

[1] Description and Operation (Vol. 1 of 1607 Magnetic Tape System, Control Data Corporation, 3 vols. 1960, pp. 3-77.

second one after the instruction in which the interrupt occurred. However, in this case, there were three locations where a jump instruction could be located which would render this approach useless. These locations were the lower half of the word in which the interrupt occurred and both halves of the following word. A search was employed to find jumps and at each place where the program might be continued, the alteration for a jump back to the interrupt routine was made. For a detailed flow chart of this operation for the case where only one search is made, see Figure C-6.

As is obvious, a complicated flagging system had to be devised to indicate the presence of parity errors when found since action might not be taken at that time, to help the A-M be aware of its whereabouts when it finished the interrupt routine, and to determine the presence of parameters for programs to be processed in the future.

The flow charts for the A-M developed from this approach are included in this paper as Figures C-1 through C-5. The results of the work at this point were presented in the term paper[1] mentioned earlier.

This A-M was not completely de-bugged, and it is superflous at this point to point out that it was tacitly assumed that an interrupt would always come when the program was waiting for a buffer operation to be completed (if the interrupt did come when a buffer operation was in progress.) This might not be true at all, in fact the "sense parity error"

[1] Brice L. Bradshaw, op. cit.

and its associated jump to the proper action routine might well be the replaced instructions using this approach and would result in disaster.

A better approach was taken in the next phase of the development which will be taken up in the next section.

4. Development of the Auto-Monitor

The development of the A-M may be thought to have been done in three phases. The first phase has been summarized in Section Three and the flow charts, Figures C-1 through C-5 in APPENDIX C.

The second phase was concentrated on the task of writing a working interrupt routine, called Subl at this time. The first step in this phase was a review of the first phase Subl with the purpose of re-evaluating the essential features required and the means of achieving them. These were:

First: Examination of the interrupt period selected. No change appeared necessary here, for the problem had not changed and there was no evidence to suggest that this period was too short for satisfactory program execution once it was de-bugged.

Second: Parity error checks, especially on input program read-in, would have to be retained. However, it would be possible to place this feature in the interrupt routine itself, and therefore the number of cells searched could be reduced, since the instruction modification could be placed immediately after the program step at which the interrupt occurred. See Figure C-6 for this function.

Third: A flag to indicate that the interrupt routine was in progress was still considered necessary. This was so that if an interrupt occurred while in the interrupt routine, an immediate re-entry through cell 00007 could be made.

Fourth: A program flag was retained since it was thought desirable to keep some semblance of the scan concept. It was no longer possible to save time by buffering the parameters into the computer since a wait had to be made for a parity error check.

The calculated time of the first phase Subl showed it to consume 1340 usecs of available interrupt time. While this was not thought excessive, should it be reducible, so much the better. Having only one cell to search should make de-bugging easier as well as reducing the running time.

Rewriting the interrupt routine, Subl, was then undertaken. As the work progressed, tests were run to determine feasibility of the approach taken. After de-bugging the programming errors, the runs showed an incompatibility in the timing assumed for tape movements. The assumed timing had been taken from various sources, such as lectures, a perusal of the 1607 Instruction Book[1] and programming aids (cf. DECOF1, a mimeographed programming aid) of the Computer Center, U. S. Naval Postgraduate School.

This incompatibility was discovered by timing the number of interrupts, comparing this with the number of times the dummy radar routine was accomplished, and comparing the point of re-entry to the program with the point of interrupt and the addresses placed in the storage locations.

[1] Description and Operation (Vol. 1 of 1607 Magnetic Tape System, Control Data Corporation, 3 Vols. 1960

This hangup occurred in the reading of the first parameter record, so three possibilities existed:

First: The interrupt was occurring too soon after the program was started.

Second: The interrupt and dummy radar routines were taking far longer than the times calculated from the timing figures given in the programming manual.[1]

Third: The buffer input was taking far longer than the five or six msecs thought to be the read cycle time.

Tests were run for all three possibilities. A wait for interrupt was placed immediately after the program start so that parameter record input would occur immediately after an interrupt routine completion with a maximum of time available. Tests were made to compare actual program running time with those calculated and the calculated times were found to be quite accurate.[2] Timing of the input buffer was made and a

[1] Characteristics of the 1604 Computer, (Minneapolis: Control Data Corporation), pp. II-5 through II-8

[2] All timing was done with programmed loops after the validity of calculations was demonstrated. This was done for this reason by running several waiting programs, such as the dummy radar routine, for a great many cycles, timing them with the real time clock. Division of total time by the number of cycles run gave the time per cycle which was compared with the time calculated. This was always within 3% agreement.

real discrepancy discovered. Instead of five or six msecs the buffer was taking from forty to fifty msecs. Further research in the 1607 Instruction Book[1] disclosed that a total of 240 msecs delay is taken when the tape is at the load point, as it was for this initial parameter read-in. This timing test, and other facts to be related show the manual to be in error in this respect. For this reason, the Subl routine was revised by the addition of tapeservice features that served to keep the tapes off the load points by resetting them. This was accomplished by console control through the use of jump keys. Also, a "Tapelockout" feature was programmed so that tapes could be changed while the A-M was working. Completion of this routine was the end of the second phase. The Subl of this phase is flow-charted in Figures C-6 through C-8 of APPENDIX C.

Another delay of 200 msecs at the finish of each read and write tape movement is specified by the 1607 Instruction Book.[2] With this delay, no interrupt rate greater than four times per second could be used for a 160 on the same channels as the input and output tapes. The timing investigations above do not show this much delay, and further investigations with the 1607 circuit blueprints show no delay except in the case of actual tape reversal of movement. Therefore, only a 40 msec delay occurs and this only at load point.

[1] Description and Operation (Vol. 1 of 1607 Magnetic Tape System, Control Data Corporation, 3 vols. 1960 p 3-66

[2] Same as footnote [1], pp. 3-65, 3-70

In the actual system, communication will be accomplished through an Inter-Computer Connecting Device (ICD) which will be part of the Data Display Unit, DD 65.

Hence, a phase three was started. The interrupt routine, now SUBA, was written for channel diversity. The advantages of this approach are:

a. Much simpler programming.

b. Easier tape handling.

c. More constant timing for the entrance to the radar processing routine. This timing varies only from 100 to 251 usecs from the time the interrupt occurs in this program as against a variance up to 10 msecs in the phase two Sub1.

d. Parity errors are detected in completely normal fashion.

Therefore, even though the 200 msec delays are not actually incorporated in the 1607 system, it appears much more advantageous to use this simpler routine.

The phase three A-M dropped the scan concept since there is actually little to scan at this time. The assembler is in process of a study to improve it, a FORTRAN compiler is being investigated, the Library computer use has not been made firm and it seems of little value to program for such scanning with nothing known of the actual requirements of the scan.

Therefore, the SUBB and SUBD are concerned with read-in, running and dump of programs only. Their de-bugging was a straightforward process with one exception. After completion of the processing of the last program, the next parameter read-in will find only a single word, namely the second end-of-file mark which indicates the last taped program. The buffer was therefore not terminated and a shift to another tape unit was not successful. The 1607 Instruction Book[1] states that the buffer must be artificially terminated. The programming manual[1] shows an instruction (740 30000) for clearing channel three but this instruction does not terminate the buffer. Another device tried was equalizing the addresses in cell 00003 by loading an index with the lower address and storing it in the upper half word, address field. This does not terminate the buffer either. The successful device is to activate a buffer whose starting address is the same as the known address in the lower half of cell 00003.

With the SUBB and SUBD routines de-bugged and working properly, the A-M routine worked satisfactorily and the timing tests were conducted. The results of these tests to demonstrate feasibility are given in Section 2.

[1] Description and Operation (Vol. 1 of 1607 Magnetic Tape System, Control Data Corporation, 3 vols. 1960, 11 3-54

5. Conclusions and recommendations for further development

It is believed that the Auto-Monitor described in Section 2 with the results of the tests given there show that a program of this type is feasible for use in a time-share application of this type.

There remains the problem common to all monitoring programs of an operator's being unable to stop the machine at will, do trouble shooting from the console and perform other console functions. Just as the automatic feature has many advantages, this restriction may become a disadvantage. At this time it would appear that the input programs should be restricted to error free programs which regularly process large amounts of data. These programs could easily be deferred to a time when radar processing is to be done, and the two carried on simultaneously. Those programs which must have console attention will be done at other times.

The dotted blocks of Figure 1 show a logical extension of the present system. Here card readers provide the input to the computer through a 160. For this development, an assembler or compiler must be adapted for continuous use.

In its present form, the following developments are necessary for successful use:

a. The adaptation of the BCD Listable dump for continuous use.

b. The modification of an assembler and the A-M to provide compatibility for their use together.

c. A means of reading paper tape, possibly enabled by using jump keys.

The present FORTRAN has no stops and all entries are made without stopping the computer. It would appear that this is an ideal medium for future exploitation, in that all operator communication with the computer will be done through a compiler of this type. Programs will then be compiled and run in straightforward fashion.

Steps necessary for this extension are:

a. Modification of the A-M and a compiler for simultaneous use.

b. Enablement of paper tape read and possibly other inputs for use by the compiler.

c. Incorporation of a more extensive scan when the requirements are more definite.

BIBLIOGRAPHY

1. *Characteristics of the MODEL 1604 COMPUTER.* Minneapolis: Control Data Corporation, 1959, Revised 7/1/60

2. *Description and operation.* Vol. 1 of *1607 Magnetic Tape System.* 3 vols. Minneapolis: Control Data Corporation, 1960

3. *Maintenance.* Vol. 3 of *1604 Computer.* 4 Vols. Minneapolis: Control Data Corporation, 1960

4. *Principles of Operation.* Vol. 2 of *1604 Computer.* 4 Vols. Minneapolis: Control Data Corporation, 1960

5. Bradshaw, Brice L. "Auto-Monitor Routine." Unpublished term paper, U. S. Naval Postgraduate School, Monterey, 1960

APPENDIX A

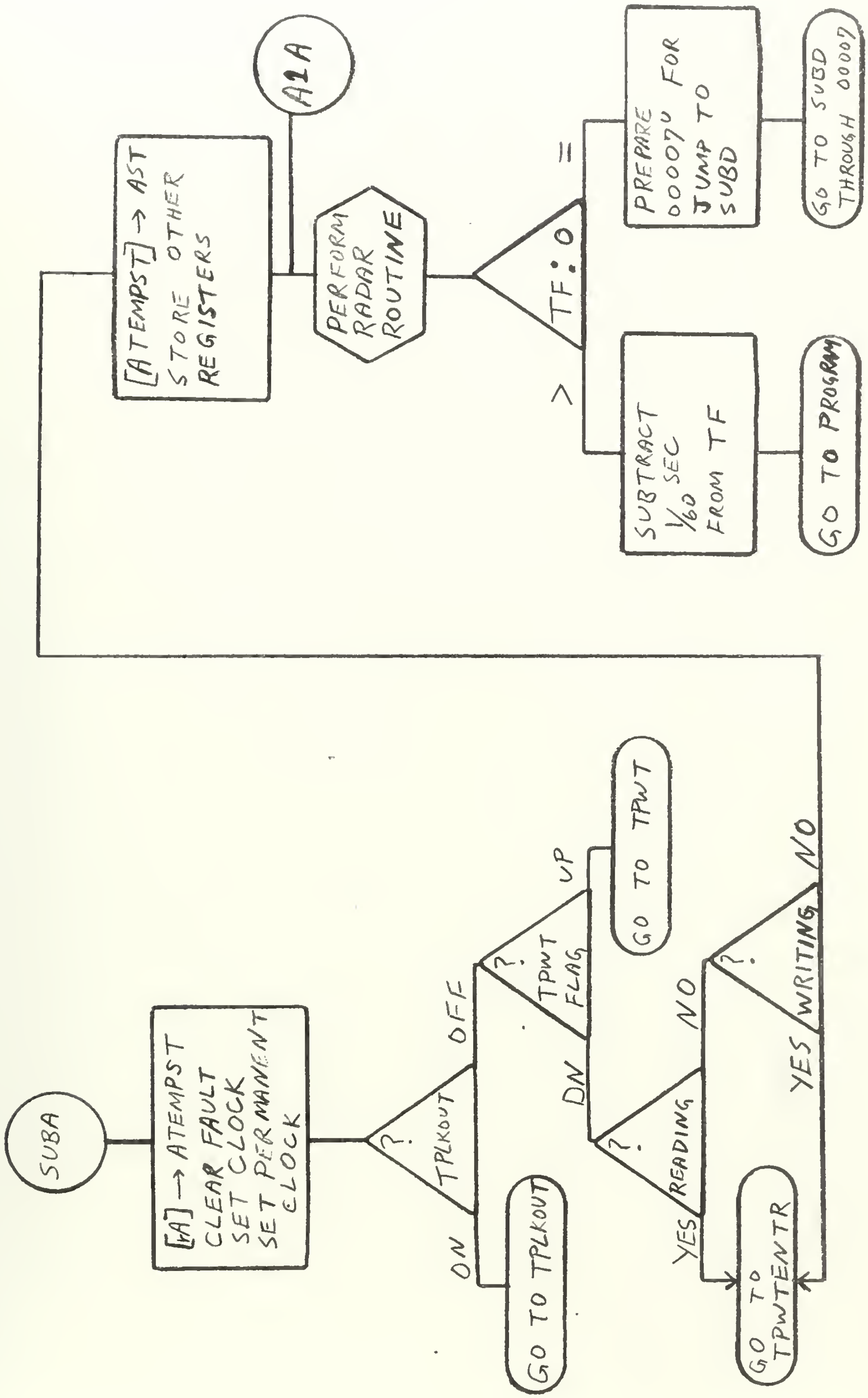


Fig. A-1 SUBA

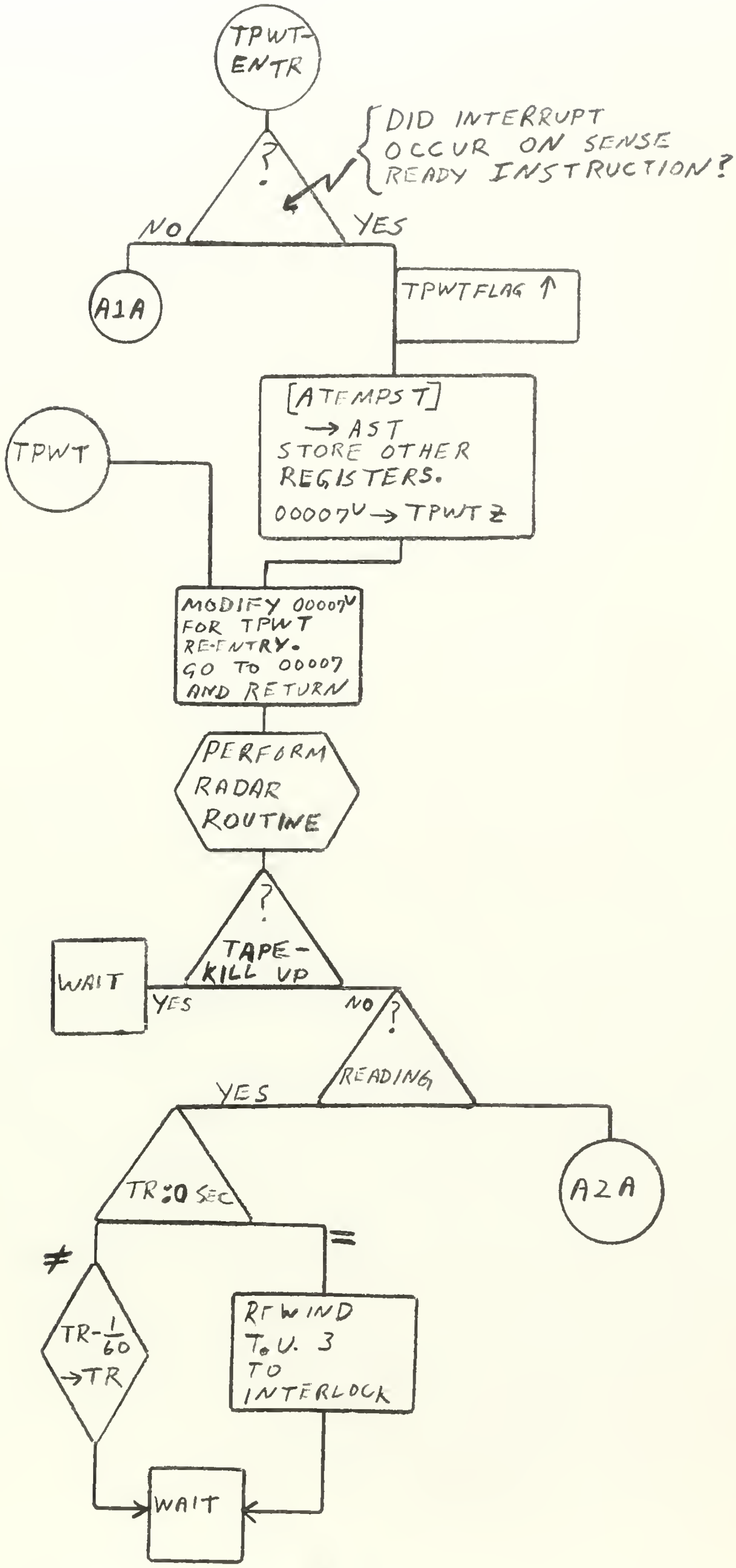


Fig. A-2 TPWT Sub-Routine of SUBA

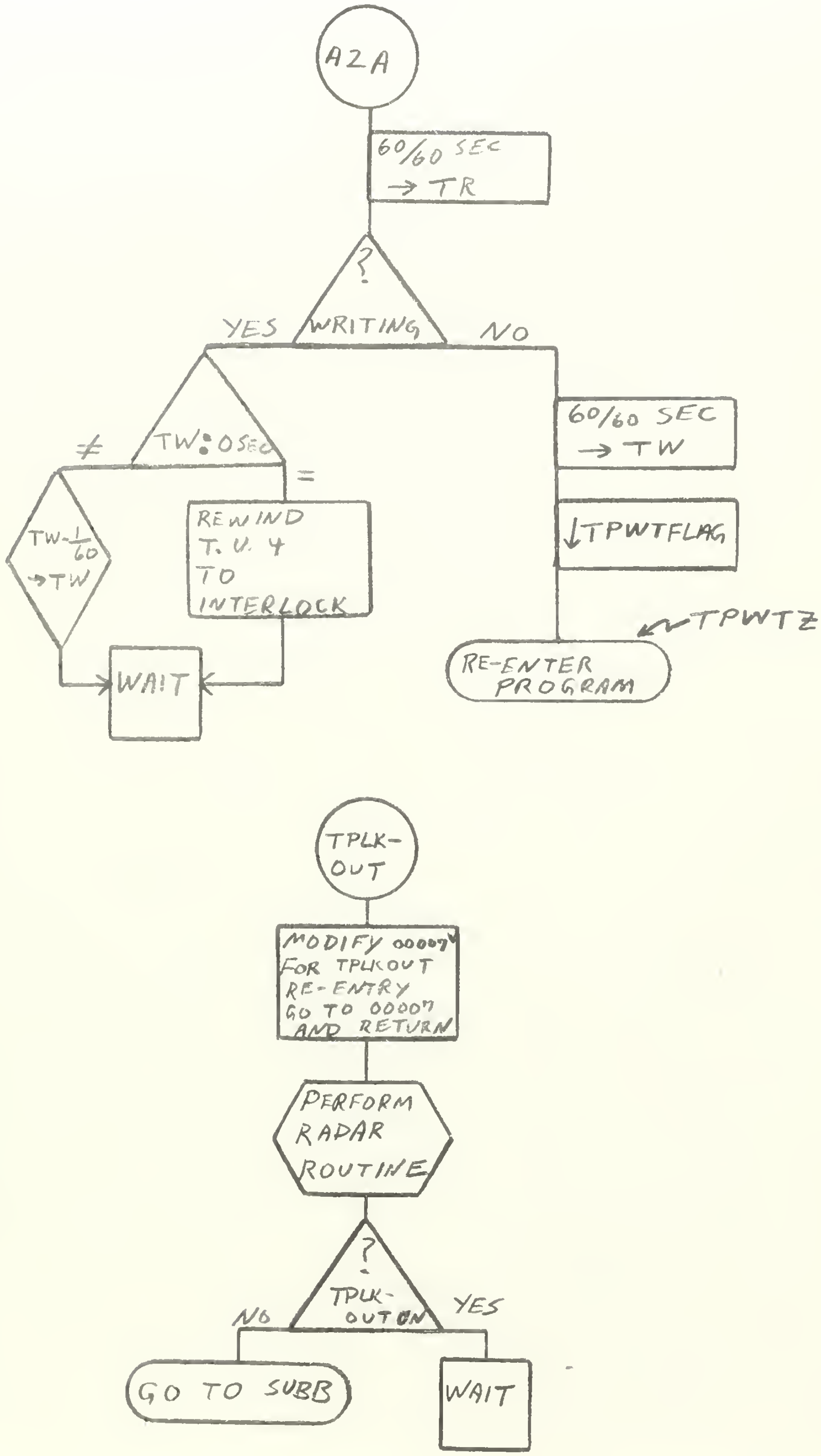


Fig. A- 2 TPWT Sub-Routine of SUBA (cont)

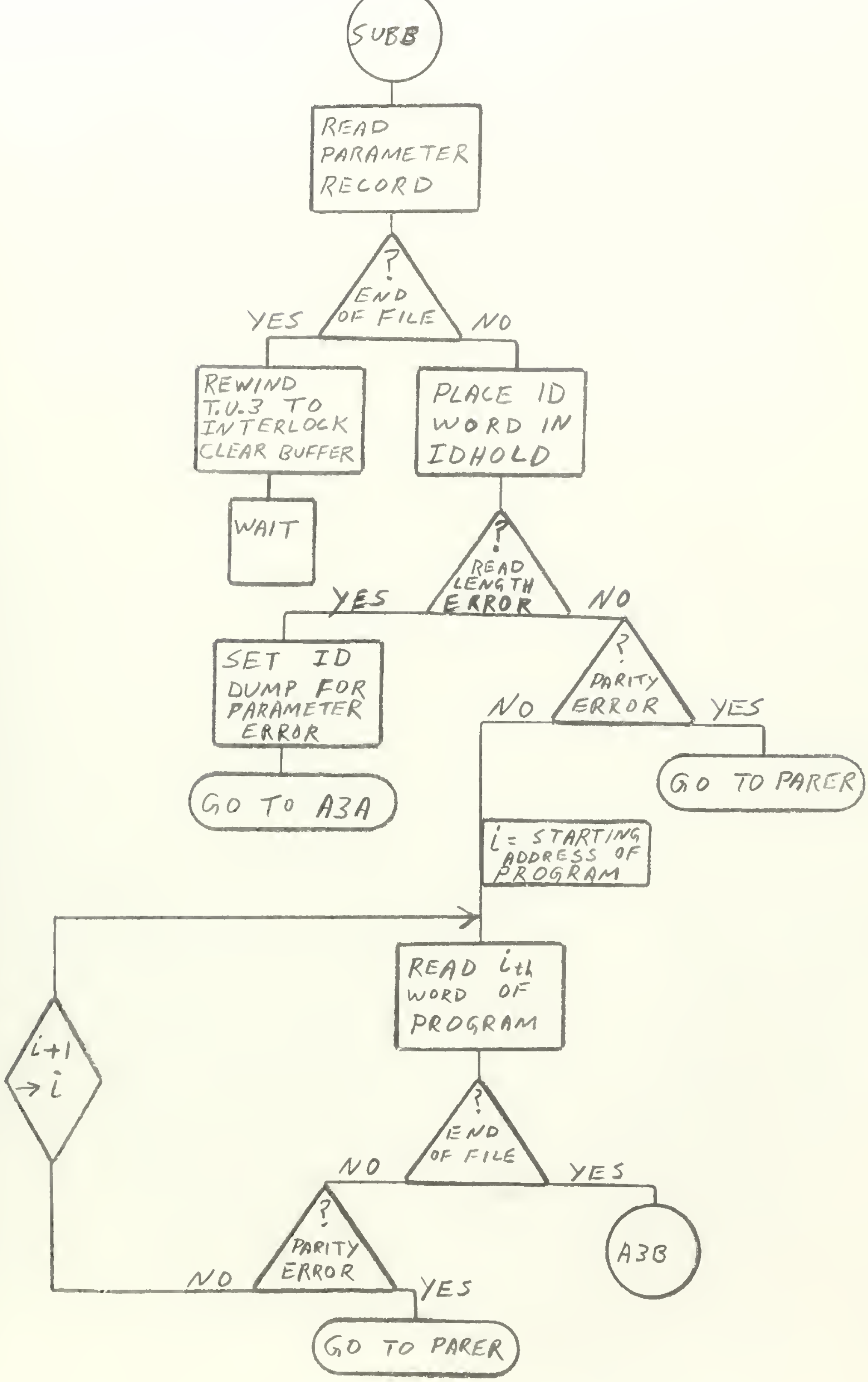
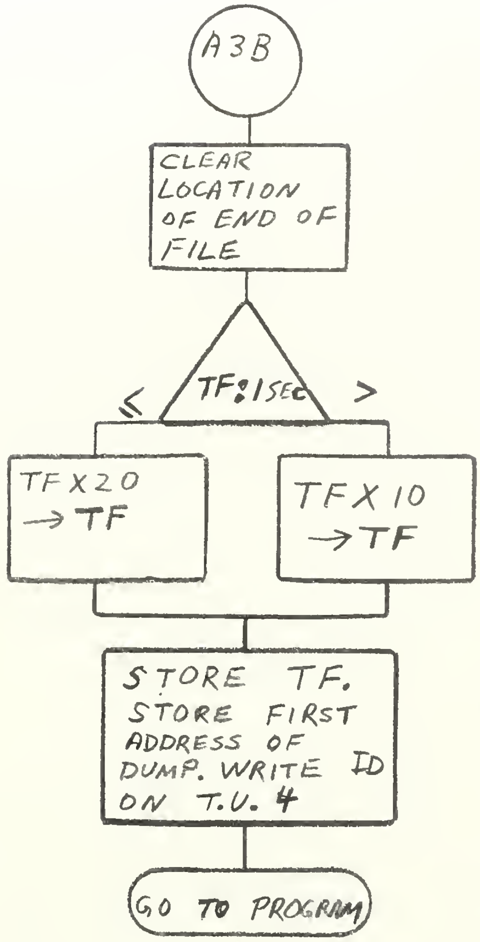


Fig. A-3 SUBB Routine



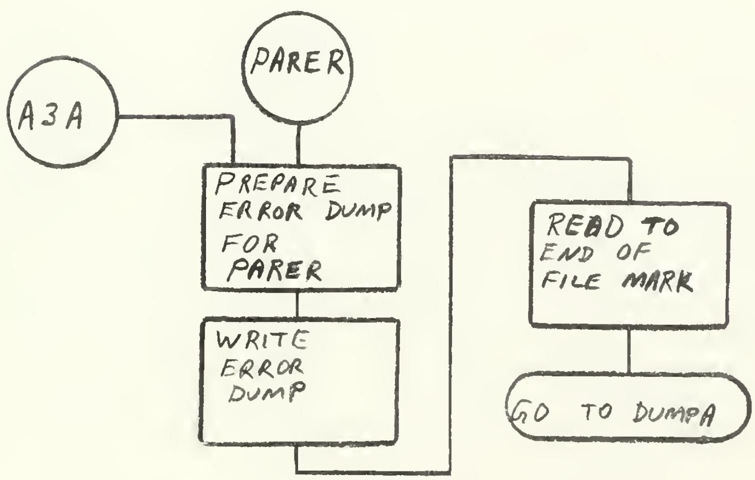


Fig. A-3 SUBB Routine (cont)

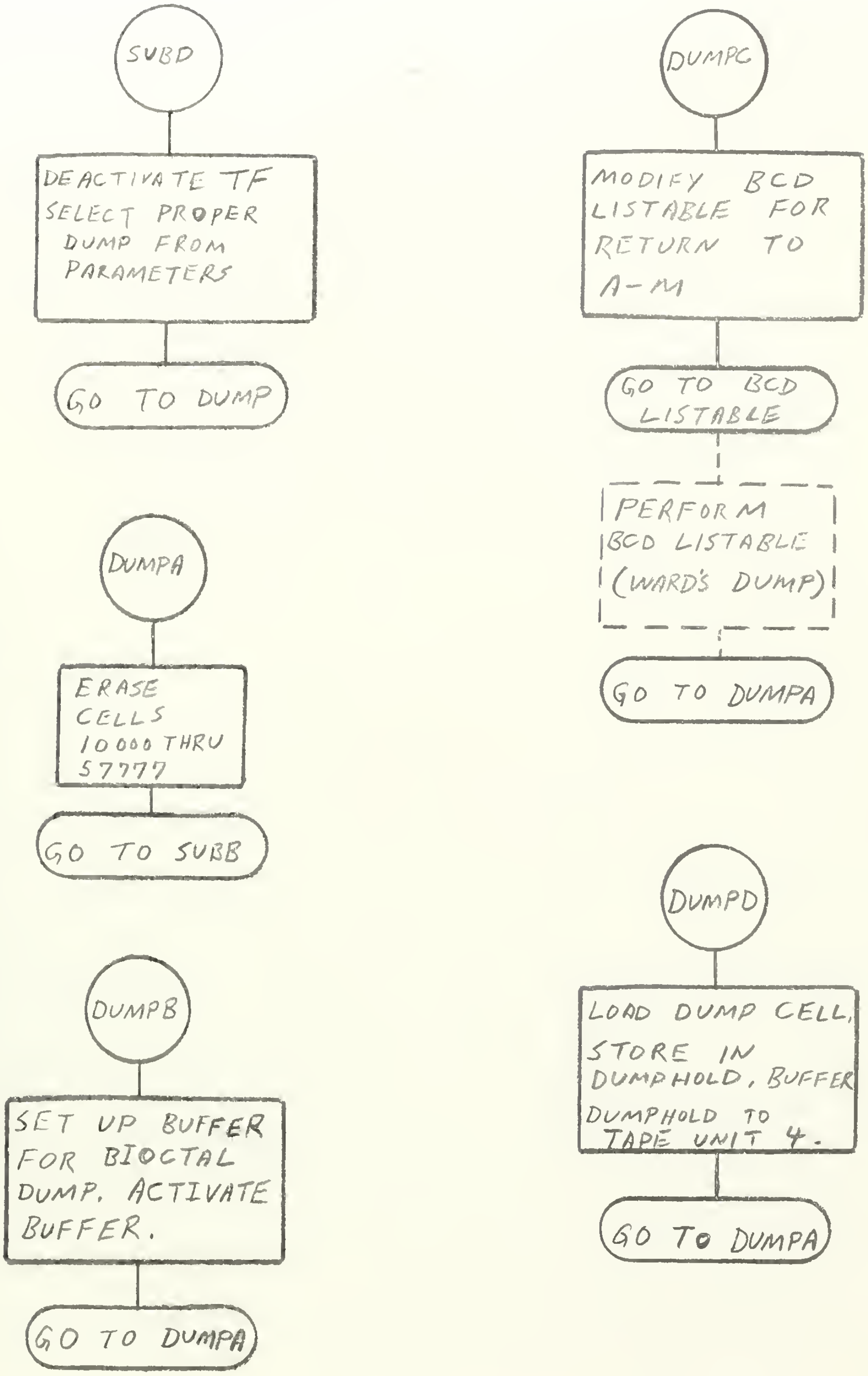


Fig. A-4 SUBD Routine

APPENDIX B

USE OF AUTO-MONITOR

1. To bootstrap A-M.

   A-M program must be on tape unit "Three." Check "Three" rewound.

   Clear the computer and step once.

   Enter 06000 in address field of A register, lower half.

   Enter 200 00003 into U register. Step once.

   Enter 740 32031 into U register. Step once.

   Enter 743 05000 into U register. Step once and clear the computer.

   Enter 05003 into Program Address Register and Start the computer. Normal stop at PAR=05013
   Error stops: PAR=05011 Parity error.
   PAR=05012 Read length error.

2. To start A-M.

   Enter 05001 into PAR and Start the computer.

3. To process program tape.

   Designate program tape "Three." Check desired dump tape is designated "Four."

   Put jump key 1 "up." This removes Tapelockout.

   Note: When the tape on "Three" has been processed and the dump completed for the last program, "Three" will rewind to interlock. Perform procedure (4.) below.

4. To change program tape, or dump tape after programs completed.

   Check "Three" rewound to interlock.

   Put jump key 1 "down."

Tapelockout is now on. Put new program tape on "Three" or redesignate as "Three" a unit having a program tape to be processed. Dump tape may be similarly changed.

Put jump key 1 "up" to restart processing.

5. To load a program to the program input tape.

Place desired tape on "Three."

Insure program is in the computer.

Enter program identification in BCD into the A register.

Enter the LIB call into Q register. (This feature not now incorporated)

Enter starting address of program into B^1. (must be first program address)

Enter last program address in B^4.

Enter first address of block to be dumped into B^2.

Enter number of locations to be dumped into B^5.

Enter dump code into B^3:

0	No dump desired
1	Bioctal dump
2	BCD Listable dump
3	Test dump (Test only)

Enter estimated program running time into B^6. Time is to be computed in seconds, entering next largest whole second, e.g., 270 msecs would be entered as 1.

Enter 05002 into the PAR. If the program is to be the final program on the tape, set jump key 3 "Up."

Start the computer.

Note: The parameters entered in the console registers will be placed in a single five word record whose composition is as follows:

Word	Upper half	Lower half
First Word	Identification, 8 BCD characters or less	
Second Word	Start address	End address
Third Word	First dump address	Number of dump words
Fourth Word	Dump code	Running time
Fifth Word	Up to 8 LIB blocks	(unavailable)

The program follows this record in one word records. User's program must contain a jump to 05252 upon its completion to assure continuity of the A-M.

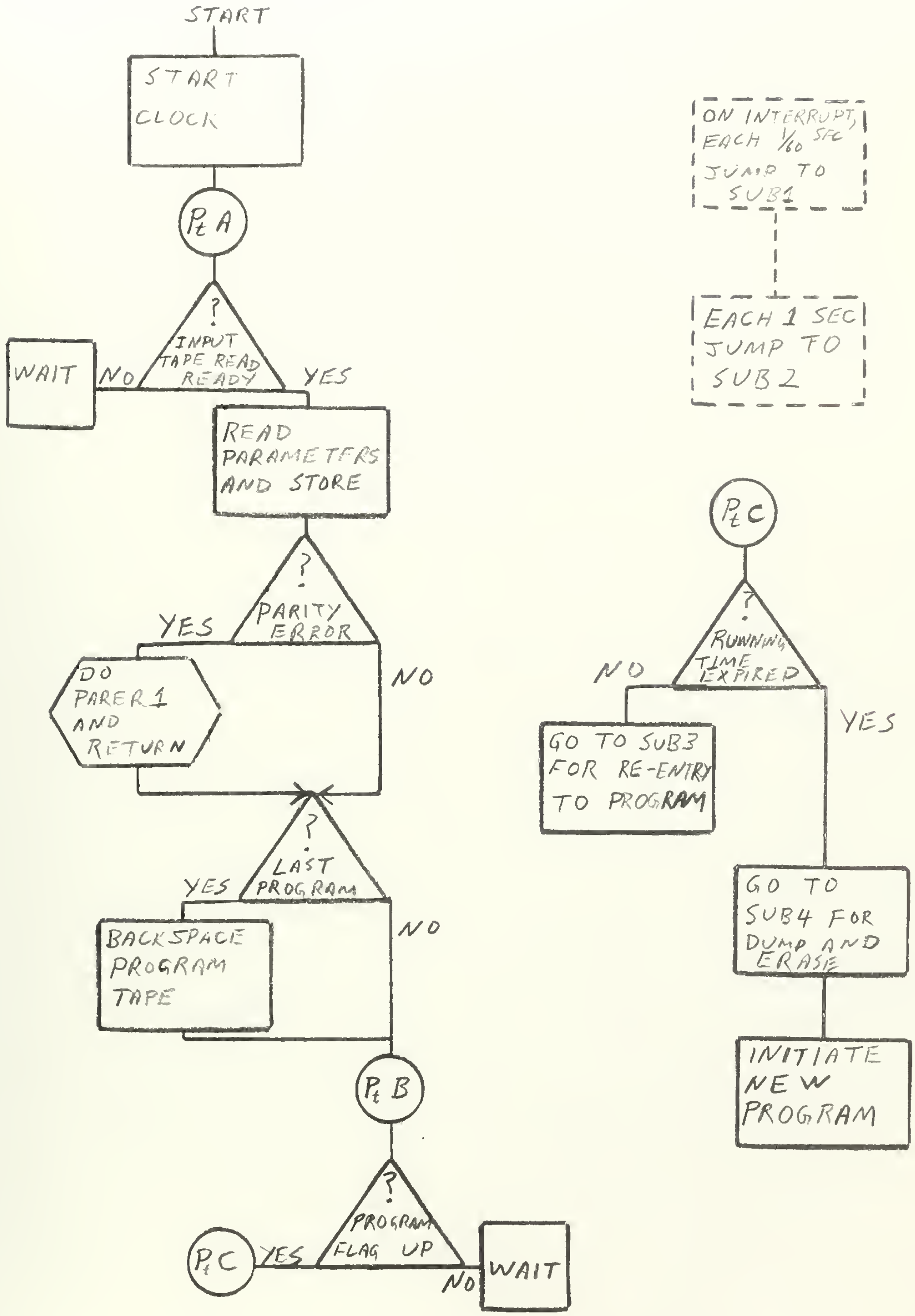


Fig. C1 - First Phase Block Diagram

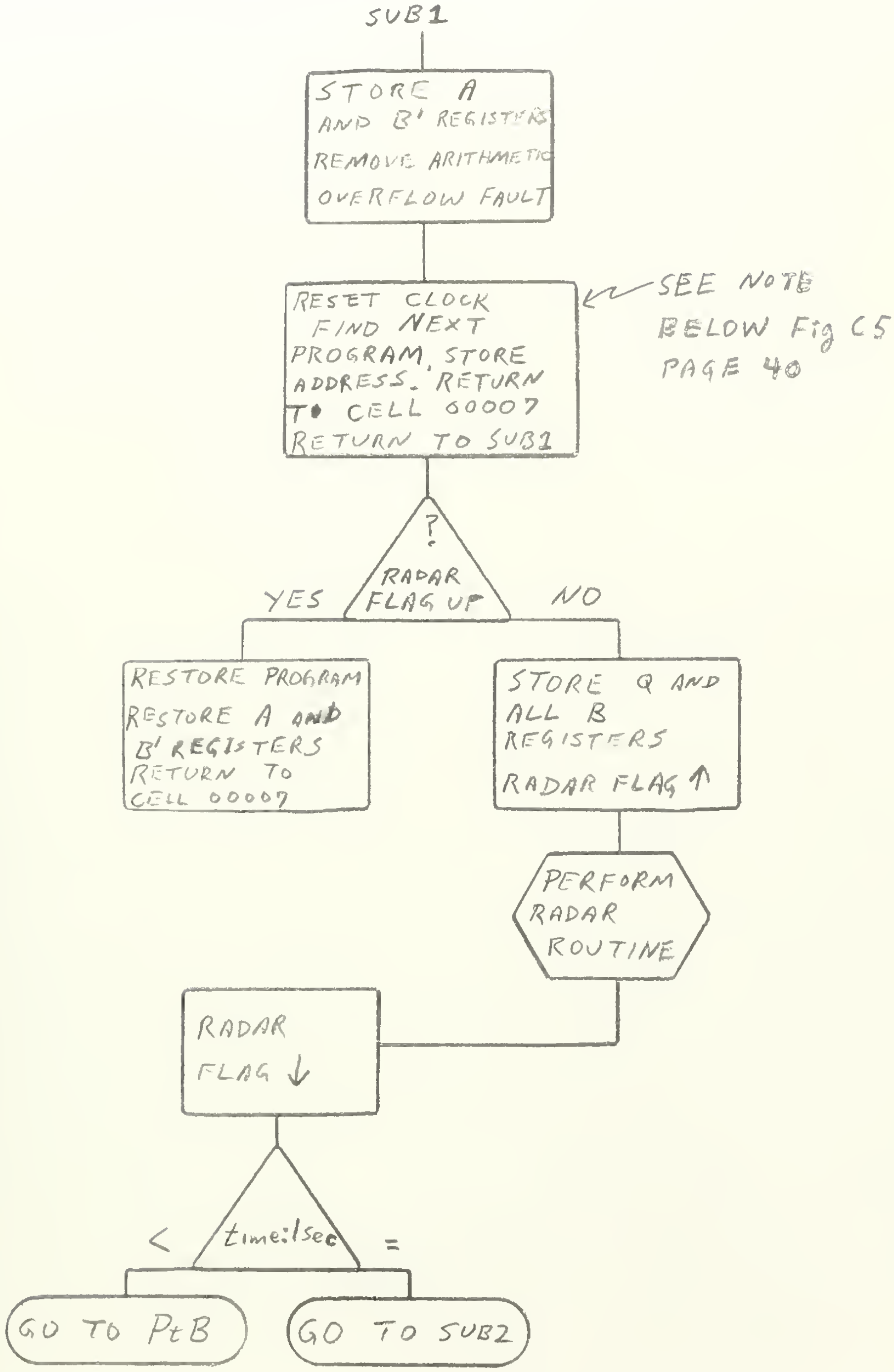


Fig. C2 - First Phase Sub1

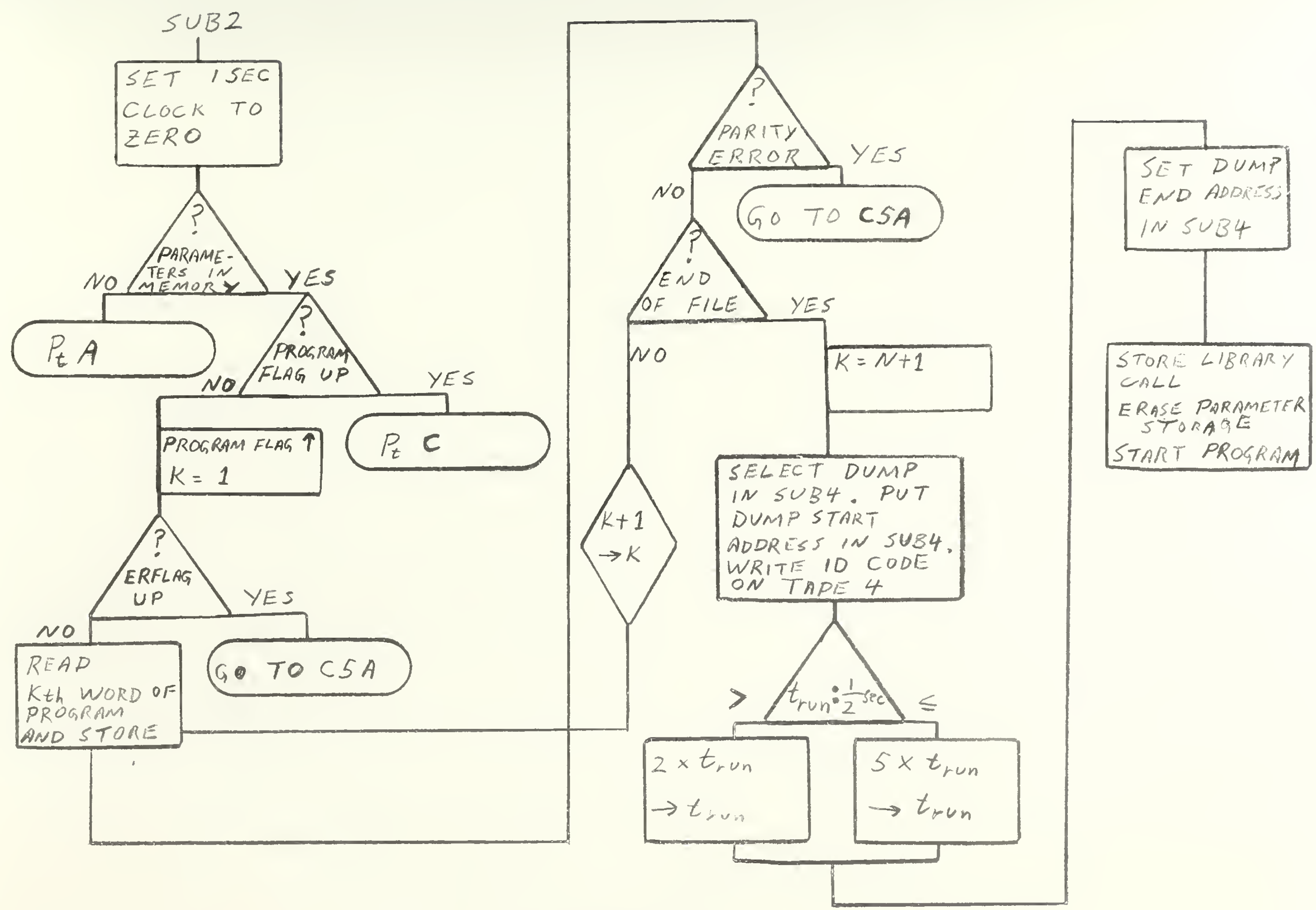


Fig. C3 - First Phase Sub2

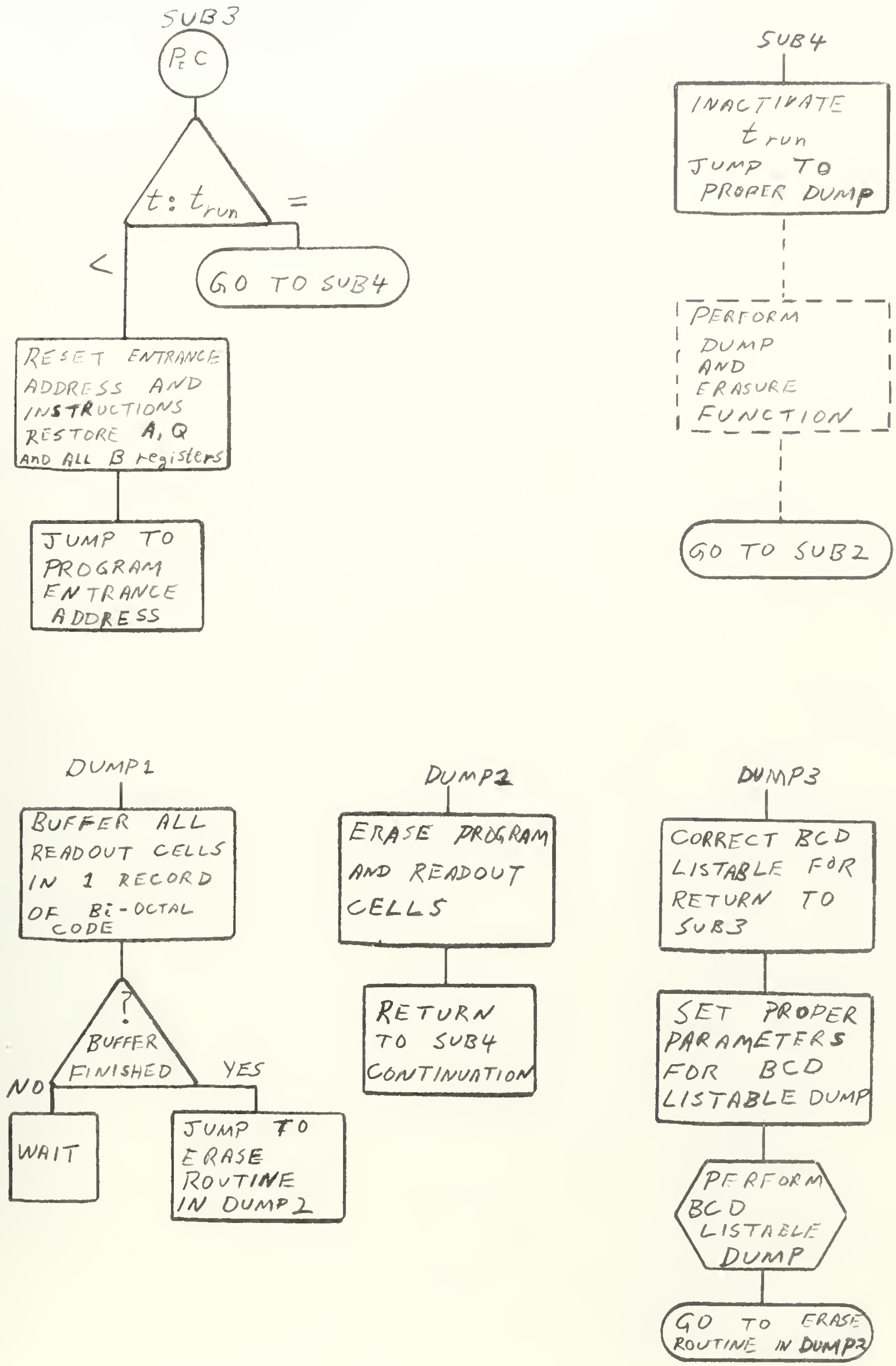


Fig. C4 - First Phase Sub3 and Sub4

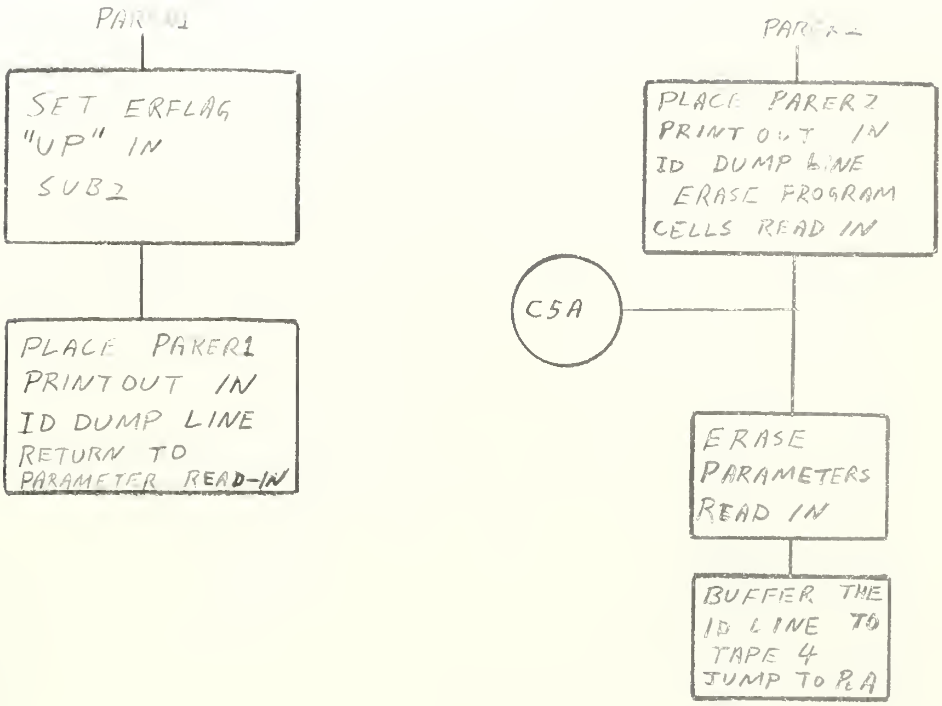


Fig. C5 - First Phase Parity Error Routines

Note: This block contains the search of the lower half of the instruction word in which the interrupt occurs and the following word, both instructions. A jump back to the auxiliary routine must be placed at each possible address to which the program may jump. Cell 00007 is then entered.

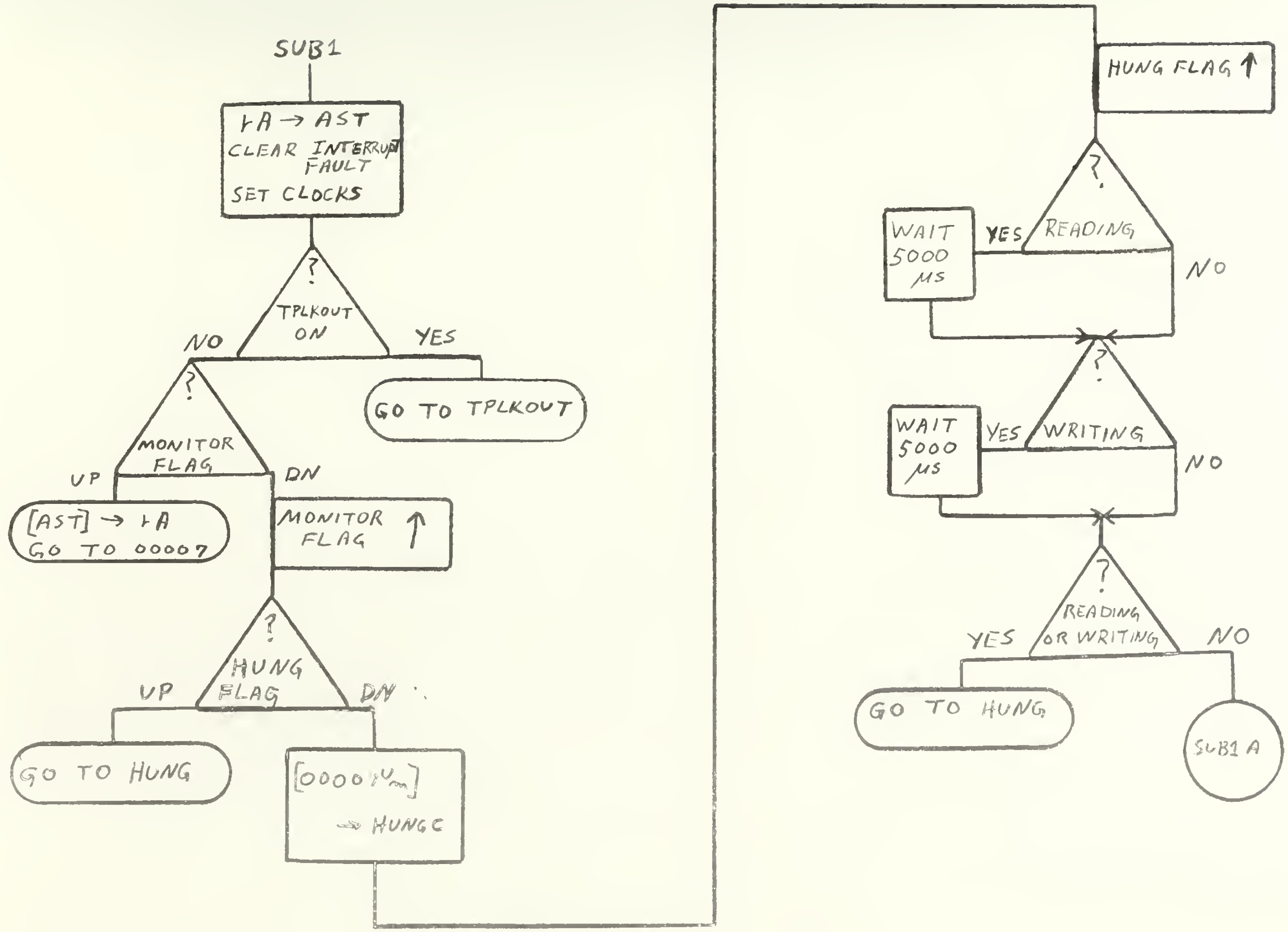


Fig. C6 - Second Phase Sub1

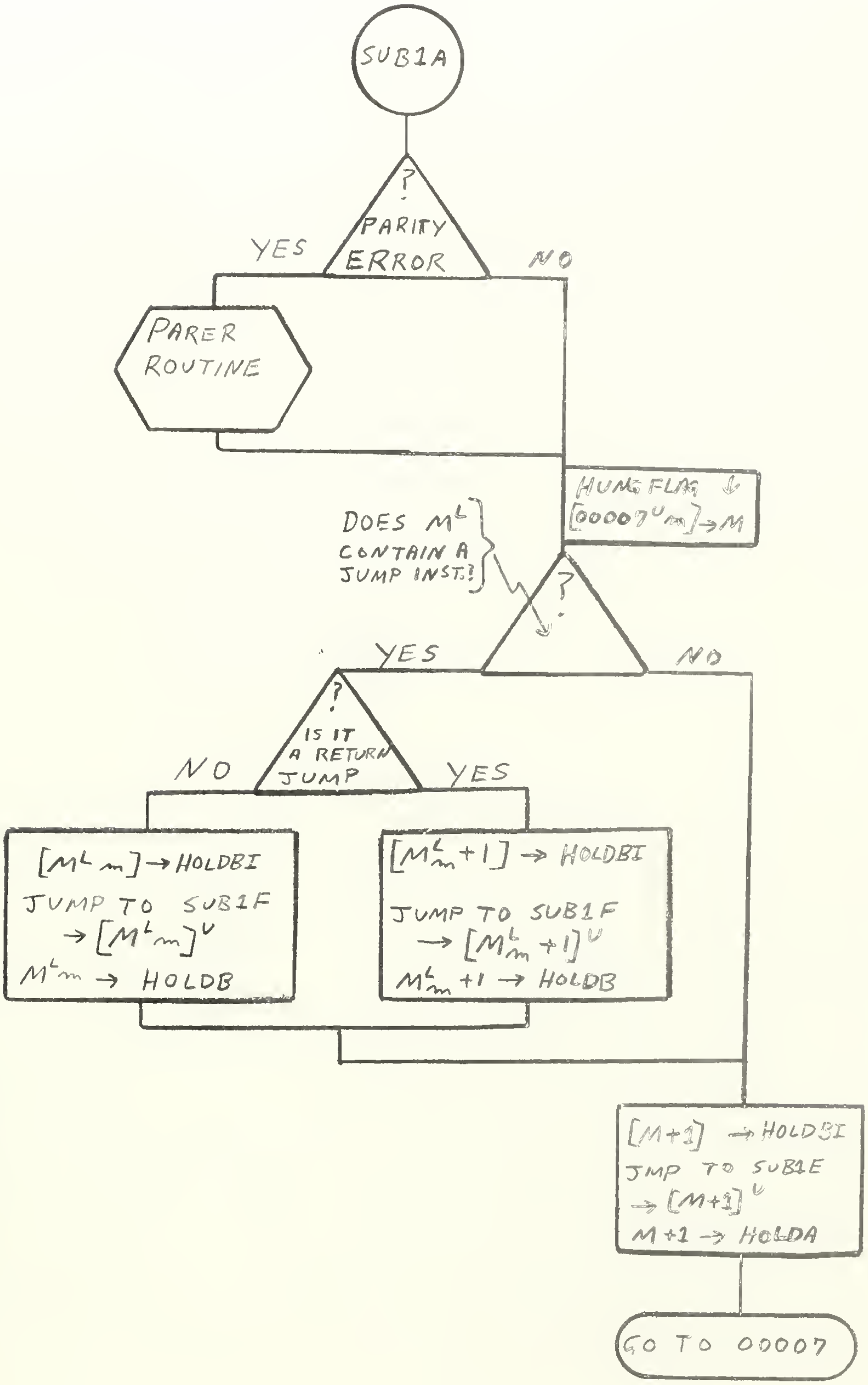

SUB1A
PARITY ERROR
YES
NO
PARER ROUTINE
HUNG FLAG ↓
[00007ᵁm]→M
DOES Mᴸ CONTAIN A JUMP INST.?
YES
NO
IS IT A RETURN JUMP
NO
YES
[Mᴸm]→HOLDBI
JUMP TO SUB1F
→[Mᴸm]ᵁ
Mᴸm → HOLDB
[Mᴸm+1] → HOLDBI
JUMP TO SUB1F
→[Mᴸm+1]ᵁ
Mᴸm+1 → HOLDB
[M+1] → HOLDBI
JMP TO SUB1E
→[M+1]ᵁ
M+1 → HOLDA
GO TO 00007

Fig. C6 (cont)

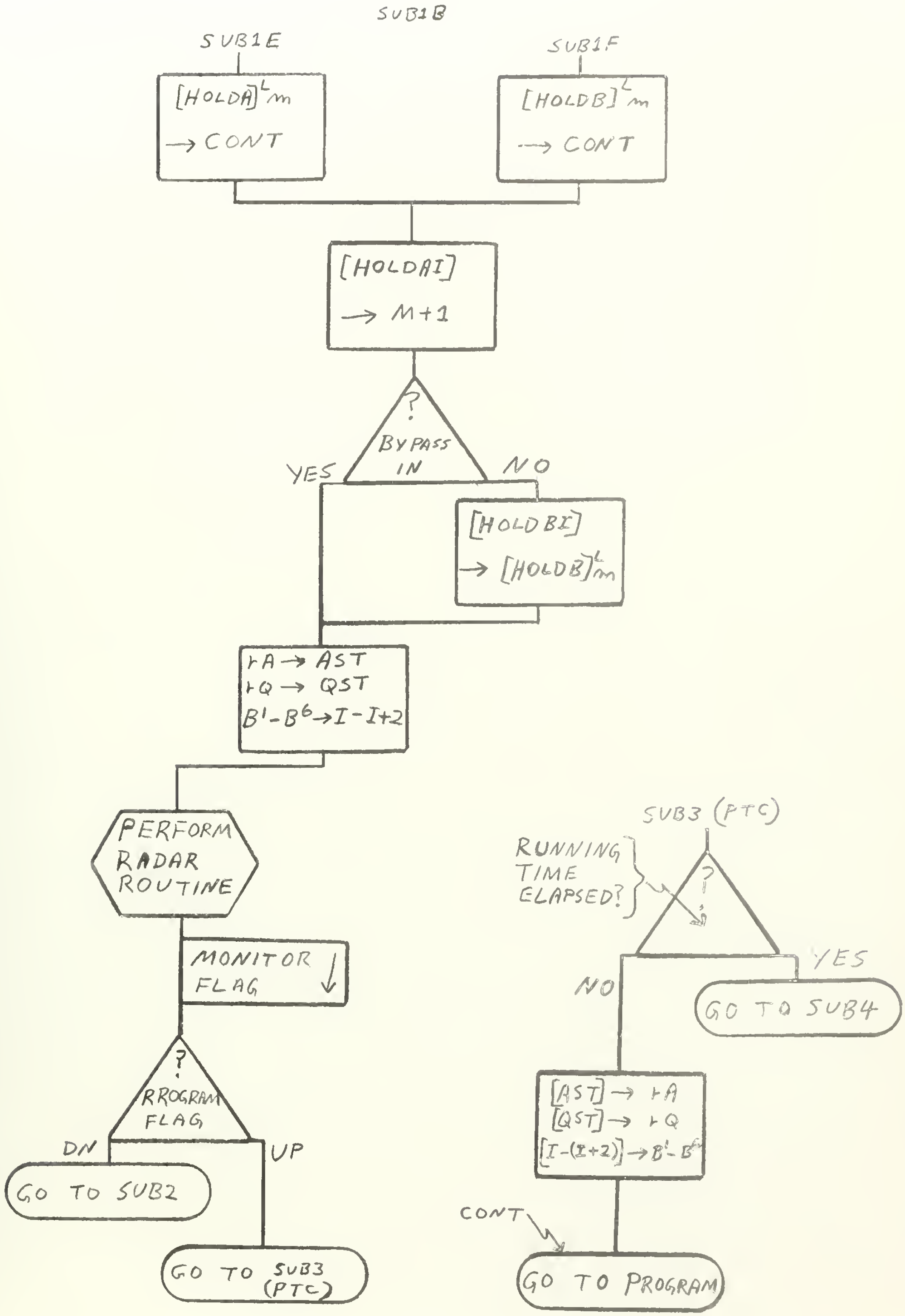

SUB1B
SUB1E
SUB1F
[HOLDA]^L_m → CONT
[HOLDB]^L_m → CONT
[HOLDAI] → M+1
? BYPASS IN
YES
NO
[HOLDBI] → [HOLDB]^L_m
+A → AST
+Q → QST
$B^1 - B^6$ → I - I+2
PERFORM RADAR ROUTINE
MONITOR FLAG ↓
? PROGRAM FLAG
DN
UP
GO TO SUB2
GO TO SUB3 (PTC)
SUB3 (PTC)
RUNNING TIME ELAPSED?
YES
NO
GO TO SUB4
[AST] → +A
[QST] → +Q
[I-(I+2)] → $B^1 - B^6$
CONT
GO TO PROGRAM

Fig. C6 (cont)

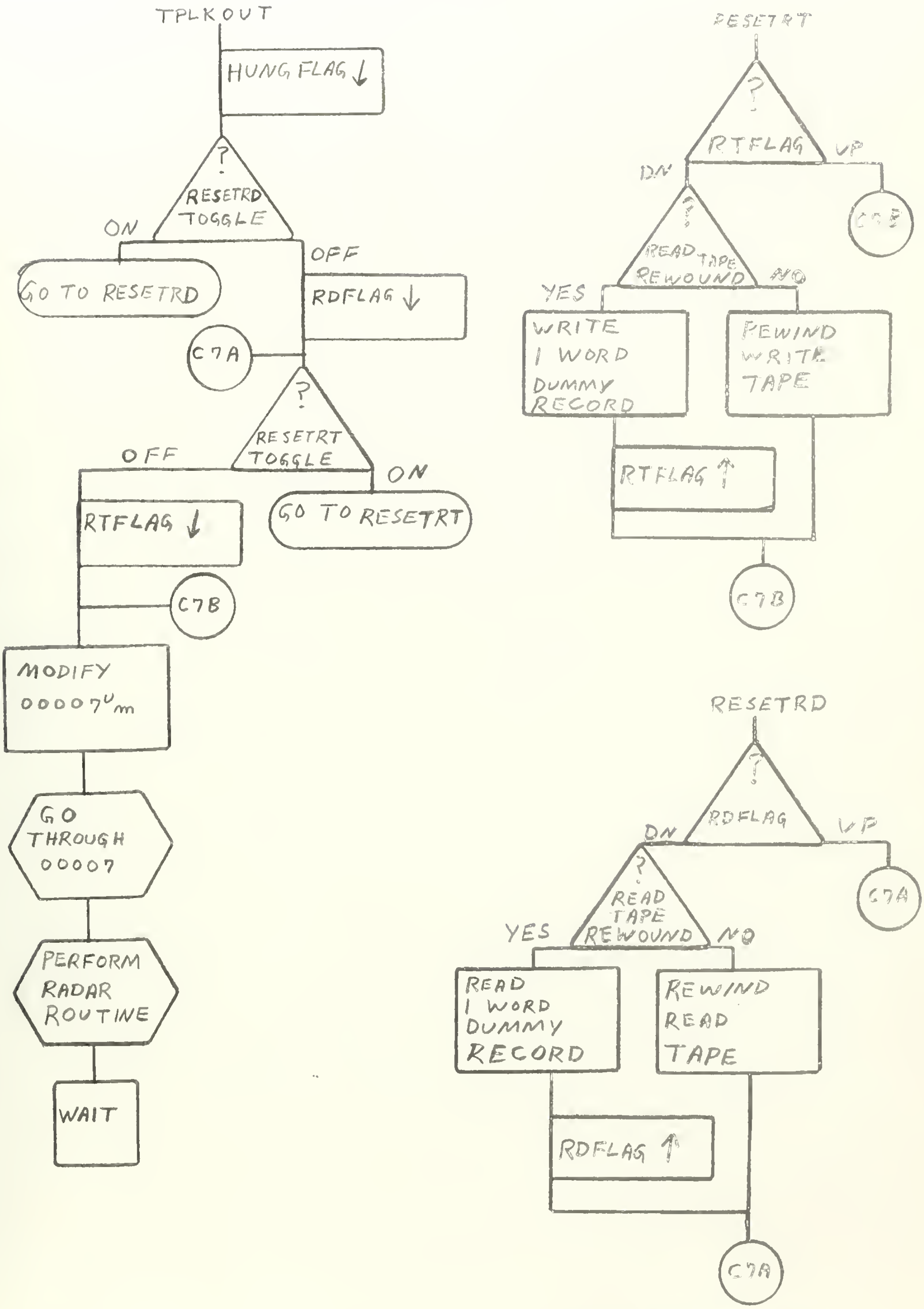


Fig. C7 - Tape Service Sub-Routines of Second Phase Subl

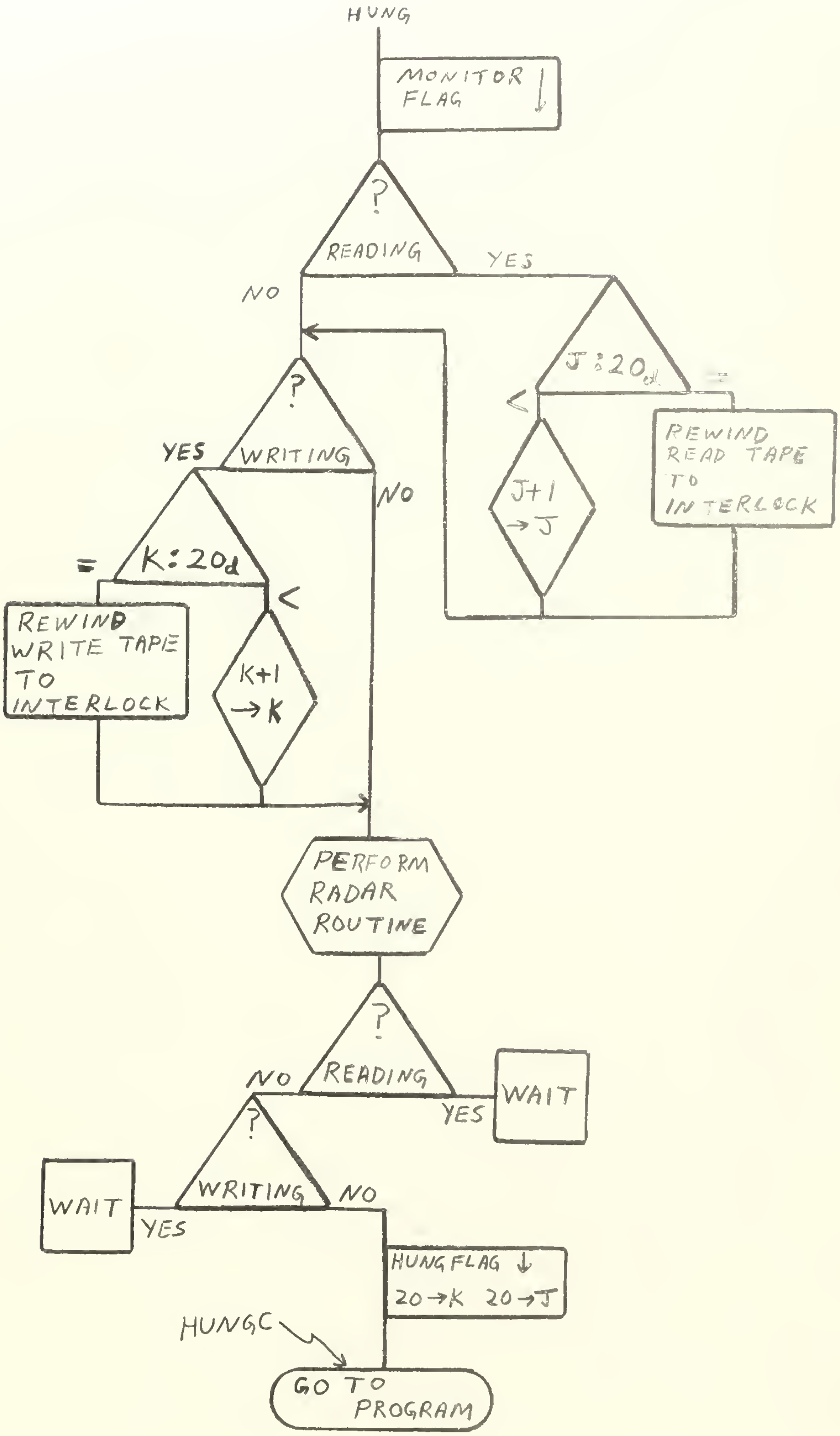


Fig. C8 - Hung Sub-Routine of Second Phase Subl